High Performance Computing with Aeronautics and Space Applications

Dr.-Ing. Achim Basermann
German Aerospace Center (DLR)
Simulation and Software Technology
Department High-Performance Computing
Linder Höhe, Cologne, Germany
DLR
German Aerospace Center

- Research Institution
- Space Agency
- Project Management Agency
Approx. 8000 employees across 33 institutes and facilities at 20 sites.

DLR Institute Simulation and Software Technology
Scientific Themes and Working Groups

Departments

Intelligent and Distributed Systems
High-Performance Computing
Software for Space Systems and Interactive Visualization

Working Groups

Distributed Software Systems
Software Engineering
Data Science
Algorithms and Data Structures
Parallelization for new HW arch.
Efficient Data Processing
Embedded Systems
Modeling and Simulation
Scientific Visualization
3D Interaction
High Performance Computing – Survey of Topics

Parallel algorithms and data structures
- Numerical libraries
- Optimization algorithms and tools
- Algorithms for quantum computers

Parallelization techniques for modern computer architectures
- Parallel programming
- Programming of quantum computers
- Parallel basic libraries
- Tools for parallel software systems

Data Science
- Pre-processing, e.g. partitioning
- High performance data analytics
- Parallel monitoring
Projects in High Performance Computing at DLR
Survey on Projects

- Exascale computing
  - Solvers for quantum physics problems
  - Pre- and Processing

- Helicopter and aircraft simulation

- Optimization
  - Thermal management for spacecraft
  - Aeronautic and space problems for adiabatic quantum annealers

- HPC for Space Situational Awareness (SSA)
ESSEX Motivation: Requirements for Exascale

**Hardware**
- Fault tolerance
- Energy efficiency
- New levels of parallelism

**Quantum Physics Applications**
- Extremely large sparse matrices: eigenvalues, spectral properties, time evolution

**ESSEX**

**Exascale Sparse Solver Repository (ESSR)**
- ghost / PHIST

**ESSEX applications:**
- Graphene,
- topological insulators,
  ...

**FT concepts, programming for extreme parallelism**

**Sparse eigensolvers, preconditioners, spectral methods**

**Quantum physics / chemistry**
ESSEX: Physical Motivation and Sparse Eigenvalue problem

Solve large sparse eigenvalue problem

$$H \ x = \ \lambda \ x$$

$$i\hbar \frac{\partial}{\partial t} \psi(\vec{r}, t) = H \psi(\vec{r}, t)$$
ESSEX Software Development: Basics

- **Git** for distributed software development

- **Merge-request workflow** for code review; changes only in branches

- Own MPI extension for **Google Test**

- Realization of **continuous-integration** with Jenkins server
The ESSEX Software Infrastructure: Kernel Library (General Hybrid and Optimized Sparse Toolkit) provides

- intelligent resource management for heterogenous systems
  - automatic pinning of threads to cores
  - asynchronous execution of (larger) tasks
- some fully optimized kernels for sparse matrix methods
  - sparse matrix-(multi)vector multiplication (spM(M)VM)
  - ‘tall and skinny’ matrices in row or column major ordering
- target platforms right now: Intel CPUs, Xeon Phi and Nvidia GPUs
- programming model: ‘MPI+X’, with X=SIMP intrinsics, OpenMP and CUDA
The ESSEX Software Infrastructure: MPI + X with GHOST

- System with multiple CPUs (NUMA domains) and GPUs
The ESSEX Software Infrastructure: MPI + X with GHOST

- System with multiple CPUs (NUMA domains) and GPUs
- -np 1: use entire CPU
The ESSEX Software Infrastructure: MPI + X with Ghost

- System with multiple CPUs (NUMA domains) and GPUs
- -np 1: use entire CPU
- -np 2: use CPU and first GPU
The ESSEX Software Infrastructure: MPI + X with **GHOST**

- System with multiple CPUs (NUMA domains) and GPUs
- `-np 1`: use entire CPU
- `-np 2`: use CPU and first GPU
- `-np 3`: use CPU and both GPUs
The ESSEX Software Infrastructure: MPI + X with

- System with multiple CPUs (NUMA domains) and GPUs
  - `np 1`: use entire CPU
  - `np 2`: use CPU and first GPU
  - `np 3`: use CPU and both GPUs
  - `np 4`: use one process per socket and one for each GPU

**Option**: distribute problem according to memory bandwidth measured
The ESSEX Software Infrastructure: PHIST for Implementing Iterative Solvers

a Pipelined Hybrid-parallel Iterative Solver Toolkit

- facilitate algorithm development using **GHUST**
- holistic performance engineering
- portability and interoperability
The ESSEX Software Infrastructure: Test-Driven Algorithm Development

- new algorithm
  - Algorithms
    - implement template
    - missing kernels
    - add unit tests
    - optimize numerics
  - Comp. Core
    - add robust kernels
    - implement optimized version
    - evaluate overall performance

- established kernel library
- optimized kernel library
The ESSEX Software Infrastructure: PHIST for Implementing Iterative Solvers

**Useful abstraction: kernel interface**

Choose from several ‘backends’ at compile time, to

- easily use PHIST in existing applications
- perform the same run with different kernel libraries
- compare numerical accuracy and performance
- exploit unique features of a kernel library (e.g. preconditioners)

---

**PHIST „builtin“**

- CPU only
- F’03+OpenMP
- CRS format

**Trilinos**

- Various arch.
- Large C++ code base
- Adapter ca 1000 lines of code

**GHIST**

- No easy access to matrix elements

---

**Required flexibility**

**Hardware awareness**
Interoperability of PHIST and Trilinos

ESSEX project

Iterative solvers

PHIST

C Wrapper

“Can Use”

Anasazi (eigenproblems)

Belos (lin. eq. syst.)

PHIST builtin

Basic operations

PHIST

Epetra

Tpetra

GHOST
The ESSEX Software Infrastructure: PHIST for Implementing Iterative Solvers

Cool features of PHIST

Task macros
out-of-order execution of code blocks
  • overlap comm. and comp.
  • asynchronous checkpointing
  • ...

Consistent random vectors
make PHIST runs comparable
  • across platforms (CPU, GPU...)
  • across kernel libraries
  • independent of #procs, #threads

PerfCheck:
print achieved roofline performance of kernels after complete run to reveal
  • deficiencies of kernel lib
  • implementation issues of algorithm
    (strided data access etc.)

Special-purpose operations
  • fused kernels, e.g. compute \( Y = \alpha AX + \beta Y \) and \( Y^T X \)
  • highly accurate core functions, e.g.
    block orthogonalization in simulated quad precision
Application, Algorithm and Performance: Kernel Polynomial Method (KPM) – A Holistic View

• Compute **approximation to the complete eigenvalue spectrum** of large sparse matrix $A$ (with $X = I$)

$$X(\omega) = \frac{1}{N} \text{tr}[\delta(\omega-H)X] = \frac{1}{N} \sum_{n=1}^{N} \delta(\omega-E_n)\langle \psi_n, X \psi_n \rangle$$
The Kernel Polynomial Method (KPM)

Optimal performance exploit knowledge from all software layers!

Basic algorithm – Compute Chebyshev polynomials/moments:

for \( r = 0 \) to \( R - 1 \) do
  \( |v\rangle \leftarrow |\text{rand()}\rangle \)
  Initialization steps and computation of \( \eta_0, \eta_1 \)
  for \( m = 1 \) to \( M/2 \) do
    swap\((|w\rangle, |v\rangle)\)
    \( |u\rangle \leftarrow H|v\rangle \)
    \( |u\rangle \leftarrow |u\rangle - b|v\rangle \)
    \( |w\rangle \leftarrow -|w\rangle \)
    \( |w\rangle \leftarrow |w\rangle + 2a|u\rangle \)
    \( \eta_{2m} \leftarrow \langle v|v\rangle \)
    \( \eta_{2m+1} \leftarrow \langle w|v\rangle \)
  end for
end for

Application: Loop over random initial states
Algorithm: Loop over moments

Building blocks:
(Sparse) linear algebra library

- spmv() Sparse matrix vector multiply
- axpy() Scaled vector addition
- scal() Vector scale
- axpy() Scaled vector addition
- nrm2() Vector norm
- dot() Dot Product

Application:
Loop over random initial states
Algorithm:
Loop over moments

Building blocks:
(Sparse) linear algebra library

- spmv() Sparse matrix vector multiply
- axpy() Scaled vector addition
- scal() Vector scale
- axpy() Scaled vector addition
- nrm2() Vector norm
- dot() Dot Product
The Kernel Polynomial Method (KPM)

Optimal performance exploit knowledge from all software layers!

Basic algorithm – Compute Chebyshev polynomials/moments:

```plaintext
for r = 0 to R - 1 do
    |y⟩ ← |rand()
    Initialization steps and computation of η₀, η₁
    for m = 1 to M/2 do
        swap(|w⟩, |y⟩)
        |u⟩ ← H|v⟩
        |u⟩ ← |u⟩ - b|v⟩
        |w⟩ ← -|w⟩
        |w⟩ ← |w⟩ + 2a|u⟩
        η₂m ← ⟨v|v⟩
        η₂m+1 ← ⟨w|v⟩
    end for
end for
```

```plaintext
for r = 0 to R - 1 do
    |y⟩ ← |rand()
    Initialization steps and computation of η₀, η₁
    for m = 1 to M/2 do
        swap(|w⟩, |y⟩)
        |w⟩ = 2a(H - bI)|v⟩ - |w⟩ &
        η₂m = ⟨v|v⟩ &
        η₂m+1 = ⟨w|v⟩ &
        aug_spmv()
    end for
```

Augmented Sparse Matrix Vector Multiply
The Kernel Polynomial Method (KPM)

Optimal performance exploit knowledge from all software layers!

Basic algorithm – Compute Cheyshev polynomials/moments:

```
for r = 0 to R - 1 do
    |v⟩ ← |rand⟩
    Initialization steps and computation of η₀, η₁
for m = 1 to M/2 do
    swap(|w⟩, |v⟩)
    |w⟩ = 2α(H - b₁)|v⟩ - |w⟩ &
    η₂m = ⟨v|v⟩ &
    η₂m+1 = ⟨w|v⟩
end for

|V⟩ := |v⟩₀..R-1
|W⟩ := |w⟩₀..R-1
|V⟩ ← |rand⟩
Initialization steps and computation of μ₀, μ₁
for m = 1 to M/2 do
    swap(|W⟩, |V⟩)
    |W⟩ = 2α(H - b₁)|V⟩ - |W⟩ &
    η₂m[] = ⟨V|V⟩ &
    η₂m+1[] = ⟨W|V⟩
end for
```

Augmented Sparse Matrix
Multiple Vector Multiply
KPM: Heterogenous Node Performance

Performance in Gflop/s

- **SNB**
- **K20X**
- **SNB+K20X**

Intel Xeon E5-2670 (SNB)
NVIDIA K20X

- Topological Insulator Application
- Double complex computations
- Data parallel static workload distribution
KPM: Large Scale Heterogenous Node Performance

Performance Engineering of the Kernel Polynomial Method on Large-Scale CPU-GPU Systems

*Thanks to CSCS/T. Schulthess for granting access and compute time
EC Project CRESTA (2011-2014)

• Three year EU-funded collaborative project, 13 partners, €12 million costs, €8.5 million funding, start: October 2011
  • Collaborative Research into Exascale Systemware, Tools and Applications
    • Project coordinator: EPCC at The University of Edinburgh

• CRESTA has a very strong focus on exascale software challenges

• Uses a co-design model of applications with exascale potential interacting with systemware and tools activities

• The hardware partner is Cray

• Applications represent broad spectrum from science and engineering

• CRESTA will compare and contrast incremental and disruptive solutions to Exascale challenges
SC: Concepts for Exascale Pre- and Post-Processing

Goals: development of pre- and post-processing as well as remote hybrid rendering tools in order to support exascale applications and users with focus on

- mesh analysis and partitioning tools,
- visualization tools and
- data management tools

in a co-design process with CRESTA applications
PPSTee: A Pre-Processing Interface for Steering Exascale Simulations

- Swappable partitioning tools (ParMETIS, PTScotch, Zoltan)
- Consideration of different simulation phases like core computations and visualization
PPSTee Experiments in HemeLB

HemeLB - PPStee - aneurysm_0.025mm - Partitioning only

HemeLB - PPStee - aneurysm_0.025mm - Calculation only
Interactive Particle Seeding
DLR Project Free-Wake

- Developed 1994-1996 by FT-HS
  - implemented in Fortran
  - MPI-parallel

- Used by the FT-HS rotor simulation code S4

- Simulates the flow around a helicopter’s rotor

- Vortex-Lattice method
  - Discretizes complex wake structures with a set of vortex elements

- Based on experimental data (from the international HART program 1995)

- SC’s task: hybrid Free-Wake parallelization for CPUs and GPGPUs
GPU Computing with OpenACC

```fortran
program main
    real :: a(N)
    ...
 !$acc data copyout(a)  
    ! computation on the GPU in several loops:
    ...
 !$acc parallel loop
    do i = 1, N
        a(i) = 2.5 * i
    end do
 !$acc parallel loop
    ...
 !$acc end data
    ! Now results available on the CPU
    ...
end program main
```
Port to GPGPU and Modernization

- Successfully ported the Freewake simulation to GPUs using OpenACC
  - original numerical method not modified
  - refactored & restructured a lot of code
  - results verified on CPU and GPU
- Porting complex algorithms to GPUs is difficult
  - branches in loops hurt (much more than for CPUs)
- Loop restructuring may also improve the CPU performance
  - (SIMD vectorization on modern CPUs)
- Stumbled upon several OpenACC PGI-compiler bugs (all fixed very fast)
- Freewake on a workstation with reasonable cycle times → Goal achieved!
Parallel Coupling Frameworks

FSDM: Flow Simulator Data Manager

- CAD
- Mesh
- Flight mechanics
- Structural mechanics
- Analytics

Coupling of several application components

VAST: Versatile Aeromechanic Simulation Tool

- Simulation Specification
- Process Control (Interactive/Batch)
- Preprocessing
- Initialization
- System description
- Model Configuration
- Control
- Time Integration Loop /Solver

Record states
Collect Output
Output file(s)
Analysis and Optimization of the Spaceliner Pre-Design

- Development of a hypersonic passenger spacecraft for long distance flights

- Descent should be accomplished in gliding flight

- **New research focus:** development of a hybrid structure with integrated thermal control units involving magnetohydrodynamic (MHD) effects with cooled magnets
Implementation of a Multidisciplinary Optimization Loop

- Implementation of the design as process graph in the software platform **RCE** (remote component environment) by coupling tools from different disciplines

- Problem: no derivatives available

- Up to now: use of derivative-free optimizers from toolbox **DAKOTA**

- **Our development**: new algorithm for nonlinear derivative-free constrained optimization
  - Derivative-free trust-region **SQP**-method
Adiabatic Quantum Computer

- Optimizer for quadratic unconstrained binary problems (QUBO)

\[ E = \sum_i H_i x_i \quad \text{with } x_i \in \{0, 1\} \]
Adiabatic Quantum Computer

- Optimizer for quadratic unconstrained binary problems (QUBO)

$$E = \sum_i H_i \ x_i + \sum_{i \neq j} J_{ij} \ x_i x_j$$

with $x_i \in \{0, 1\}$
Adiabatic Quantum Computer

Example:

\[ E = 5x_1 + 2x_2 - 3x_3 - x_1x_2 + 3x_2x_3 - 4x_3x_1 \]

Lowest Energy: \( E = -3 \) at \((x_1, x_2, x_3) = (0, 0, 1)\)

- Quantum systems have discrete energy levels (e.g. atom)
- Idea: Find system whose lowest energy state (ground state) correspond to the solution of the optimization problem
Adiabatic Optimization

• How do we bring the system into the final state?
• Solution: Adiabatic evolution
  1. Prepare simple initial system with known ground state
  2. Change system slowly towards the final system
Adiabatic Optimization

- How do we bring the system into the final state?
- Solution: Adiabatic evolution
  1. Prepare simple initial system with known ground state
  2. Change system slowly towards the final system

\[ H_i^0, J_{ij}^0 \quad \text{to} \quad H_i^1, J_{ij}^1 \]

**Energy**

**Time**

**Initial system**

**Final system**
Applications for Quantum Annealers

Applications

Which problems can be mapped to QUBO?

\[ E = \sum_i H_i x_i + \sum_{i \neq j} J_{ij} x_i x_j \]

with \( x_i \in \{0, 1\} \)

- All NP-Complete Problems. E.g.
  - Graph Partitioning
  - Satisfiability Problems
- Planning
  - Job-Shop Scheduling
  - Mars-Lander Operations
- Machine Learning

* Venturelli et. al. arXiv:1506.08479
** Rieffel et. al. arXiv:1407.2887
DLR Application: Satellite Scheduling

- Model
  - 3 states
  - charge, downlink, experiment
- Goal: Maximum downlink
- Constraints: energy, data storage

Schaus et. al., DLR-SC, A Continuous Verification Process in Concurrent Engineering. AIAA Space 2013
DLR-NASA Application: Air Traffic Management

- Optimization problem represented as QUBO
- Preprocessing: cluster dependent conflicts
- Computation of simplified instances
- Preliminary results:
  - Success probability decreases with increasing problem size
  - Precision of parameters limited
DLR Software BACARDI: Backend Catalog for Relational Debris Information

- Increasing number of space debris
  - 26,000 known objects > 10 cm
  - Objects > 1 cm problematic

- Current capabilities at DLR, GSOC
  - Orbit propagation
  - Collision detection
  - Observation planning and correlation

- Composition of a DLR database
  - TLE unprecise
  - Precise data restricted
Big Damage through Small Debris Particles
BACARDI Architecture Layers

HPC technology and methods for
• processing of orbit data from sensor data
• processing of correlation operations with the data base
Conclusions

• HPC technology successfully exploited in aeronautic and space applications

• Development of scalable and maintainable software crucial

• New architectures arise with specific application areas

• New problem formulations required for certain new architectures

• Development of new algorithmic approaches necessary

• High performance data analytics increasingly important
Many thanks for your attention!

Questions?

Dr.-Ing. Achim Basermann
German Aerospace Center (DLR)
Simulation and Software Technology
Department High Performance Computing

Achim.Basermann@dlr.de
http://www.DLR.de/sc